A Re-Programmable Platform for Dynamic Burn-in Test of Xilinx VirtexII 3000 FPGA for Military and Aerospace Applications

Ramin Roosta, Jet Propulsion Laboratory, California Institute of Technology, Pasadena
Xinchen Wang, Ixia Communications, Calabasas, California
Michael Sadigursky and Phil Tracton, California State University, Northridge

Abstract

Field Programmable Gate Arrays (FPGA) have played increasingly important roles in military and aerospace applications. Xilinx SRAM-based FPGAs have been extensively used in commercial applications. They have been used less frequently in space flight applications due to their susceptibility to single event upsets (SEUs). Reliability of these devices in space applications is a concern that has not been addressed. The objective of this project is to design a fully programmable hardware/software platform that allows (but is not limited to) comprehensive static/dynamic burn-in test of Virtex-II 3000 FPGAs, at speed test and SEU test. Conventional methods test very few discrete AC parameters (primarily switching) of a given integrated circuit. This approach will test any possible configuration of the FPGA and any associated performance parameters. It allows complete or partial re-programming of the FPGA and verification of the program by using read back followed by dynamic test. Designers have full control over which functional elements of the FPGA to stress. They can completely simulate all possible types of configurations/functions. Another benefit of this platform is that it allows collecting information on elevation of the junction temperature as a function of gate utilization, operating frequency and functionality. A software tool has been implemented to demonstrate the various features of the system. The software consists of three major parts: the parallel interface driver, main system procedure and a graphical user interface.

1.0 Proposed Testing Setup and Its Advantages

A number of people have attempted to test FPGAs using a modified Xilinx development board. This allows them to put their design and testing methods into practical use and acquire reasonable results. This approach actually does similar steps but does not stop there.

The goal of this project is to develop a flexible test platform for the Xilinx VirtexII that can overcome most of the common shortcomings: lack of useable IOs, configuration varieties, clock schemes, and connections.

Another goal of a desired testing hardware is the ability to adapt to major industrial test setups such as burn-in logic test, burn-in IO test, component screening tests, at speed test, as well as to configure for single-event testing for aerospace and military applications with few or no modifications.

The design of this project consists of the FPGA test programs and the testing setups, including two printed circuit boards (PCBs); the driver board and a burn-in proof UUT board, with high-quality impedance controlled connectors and cables. The two boards are called the driver board and test board, respectively, each containing a VirtexII 3000 chip (commercial version on driver board and a military/radiation hardened grade on UUT board). The driver board will connect to the host PC via the EPP parallel interface and will be self-configured using Xilinx EEPROMS. The EEPROMS will be configured through industry standard JTAG interface. Once the driver board is configured successfully, it will provide configuration streams to the UUT board via the relatively high- speed Xilinx Select Map interface, which is capable of FPGA downloading and read-back operations. Also, between the driver board and the UUT board are roughly 20 general IOs flowing in each direction. Thus 40 pins are connecting the two boards through high-quality and individually shielded impedance-controlled connectors and cables. This setup will enable the test algorithm to verify the FPGA under test with relatively high-frequency ranges. Signal integrity has been a major concern since few people have attempted this before on burn-in test setups. A BGA-728 pin socket is placed on the test board to provide mounting for the FPGA under test. On the UUT board, there are loop-back connectors that route to almost all the IOs available to the test FPGA for various IO testing. The clocking arrangement between the two boards allows multiple clock domain operations. Users can select any one of four independent oscillators or any combination of oscillators. The VirtexII digital clock manager

(DCM) phase-shift functionalities can be used to achieve the best results especially at high frequencies.

1.1 Building Blocks of the Testing Setups

The testing system can be described in the following block diagram

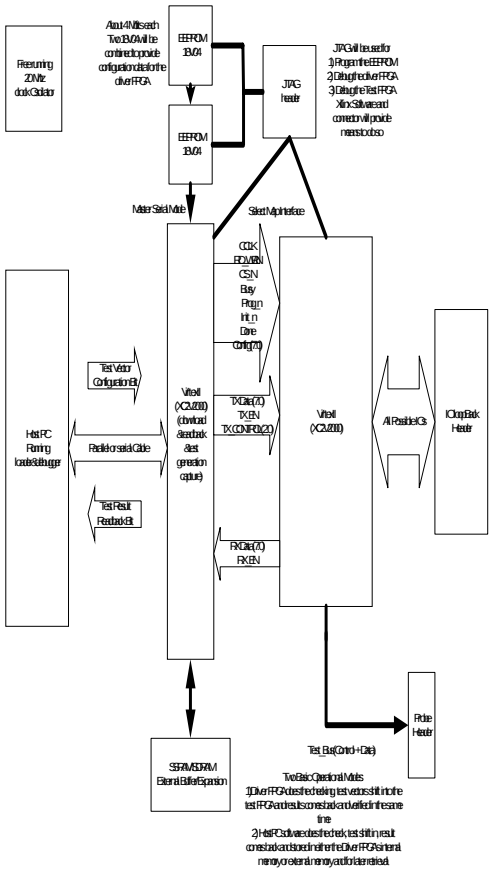

Figure 1: Burn-In Setup, System Block Diagram

The entire system consists of two boards, a driver board and a UUT board (test FPGA) and a high-speed cable to interconnect them. The driver board's roles are communicating with the host PC via EPP, setting up test vectors, read-back configuration bit streams for the test FPGA, and retrieving and comparing tests results. The UUT board's role is to host the FPGA under test, to accept test vectors, and return test results.

2.0 Description Of Major Components

The main components used on the boards are:

-Various commercial and Mil Spec oscillators.

-Three XC18V04 EPPROMs. These are Xilinx specially designed XC18V00 series In-System Programmable (ISP) Configuration PROMs for re-programming and storing various sizes of Xilinx FPGA configuration bitstreams. VirtexII FPGA XC2V3000 requires a 10,494,368-bit configuration string.

-A 728-pin socket for the test FPGA to improve signal integrity while providing 100% fan-out. The burn-in socket was placed in the center of the board. The socket allows for easy solderless replacement of UUT whenever desired. Two different burn-in sockets were used for the first revision of the UUT board. The first version used a socket manufactured by Enplas. For the second version of the board, a 728BG12B143 socket by Plastronics was used. The latter allows UUT temperature monitoring using infrared sensors and is well suited for radiation testing. Both sockets are rated for an operating temperature range of -60°C ~ +150°C.

-Three sets of high-speed, individually shielded Teflon 50-Ohm coaxial pin cables with Samtec impedance matched high-density connectors. The length of each cable is 1 meter.

There are two cable assemblies, one used for data and the second is used for program/configure the UUT.

3.0 Driver Board FPGA

The VirtexII 3000 on the driver board (driver FPGA) acts as the brain of the entire system. It is responsible for communication with the user through the PC's parallel port. It stores user-defined test vectors and sends them out to the test FPGA upon receiving a specific command. It provides necessary download/read-back protocol signals and a datastream to configure and debug the test FPGA. It also acts as the built-in-self-test (BIST) logic to determine whether the incoming results from the test FPGA contain any errors.

3.1 Internal/External Test-Vector Storage Interface

At this time the hardware does not generate random test vectors. The software or the user must supply random data as desired. It can be the same value such as commonly used AA or 55, or entirely random with any kind of distribution. Initially it will be stored in a text/data file in the Host PC's hard disk.

Once the test FPGA is downloaded and the communication between the host PC and the test FPGA is successfully established, upon the user's command the software will start to move the test vector files from the host PC's hard disk to the internal/external RAM onto the driver board. In the first release of the test platform internal block RAMs are used. The RAMs are 16 Kbits each with 8 being used. Thisresults in a total maximum test vectors size of 128 Kbits (16 Kbytes). The size of the RAM is not particularly large but sufficient to demonstrate the functionality of the hardware. The user can generate appropriate test vectors to stress (exercise) the intended components (functional blocks) inside FPGA.

The 8 block RAMs are combined into one 8-bit-wide 14-bit-deep true dual-port RAM. PortA of the RAMs will be used as read and write ports. This is used to initially set up the test vectors. Once all the addresses are being walked through, the start command is given by the user to start sending test vectors to the test FPGA. PortA's address will then be driven by another counter/state machine to act as a read port for sending the data out.

3.2 Fault-Detection Interfaces (Internal or External)

The block RAMs are configured as a true dual-port RAM, i.e., they can independently read/write on both ports with totally separate addresses. Thus, PortA can be used for sending out the test vectors to the test FPGA. Then PortB can be used for comparison with the data coming back from the test FPGA. Once the RXEN signal goes from zero to one, the driver FPGA starts reading from address zero of PortB and comparing with the RXDATA stream. This is true for the Shifter and FIFO cases, since the data should stay unchanged. Once there is any discrepancy, an error bit will be set for that clock, and the error counter will be incremented by one to record the event.

There are two sets of 8-bit counters for showing error statistics. The assumption was made that the error rate will not be big enough to overflow the 64-bit counter in the software polling interval; otherwise the counter will roll over. One counter is used to count and the other is used to latch the first counter in case the software has to update the total error count. For example, if counterA is being read, it will latch counterB's value, and counterB will be cleared to zero and start counting from the beginning. If failure occurs during the read event, then countB's initial value will be set to one and not zero like it would have been otherwise. Thus it is impossible to miss any error count. Note that the total error count is accumulated and maintained by the software. The hardware is just keeping track of the incremental values. It is entirely up to the user to start or stop the comparison. Once the RXEN goes from one to zero, the error counting and data comparison will stop. The software will still display the total error count from the last test.

The calculator's fault detection is somewhat complicated since it will have multiple CRC chains and the results for each chain are unknown until the data string is fully shifted in. In this case, the driver FPGA will know the total number of CRC chains and when data has been sent. After data ends, the driver FPGA will start to pull the results back from the test FPGA for each CRC chain. The driver FPGA will then compare the CRCs of each chain with pre-recorded values. Once a discrepancy occurs, it will use the same counter mechanism as the other two cases to report error counts.

4.0 Unit Under Test

The objective of this approach is to effectively test the UUT—VirtexII 3000. It is expected that most numerous components/building blocks of FPGA will produce more failures. Therefore, it is natural to focus the testing on LUTs and block RAMs as well as other arithmetic units. In fact, for most of the designs, these three major resource components are in high demand both in terms of quantities as well as qualities. How much they are utilized and how they are interconnected and what timing can be achieved directly dictate how well the system performs.

4.1 Interfacing the Driver FPGA and Possible Clocking Schemes

The driver FPGA is interfaced with the test FPGA via two 8-bit data buses (for each direction) plus some control bits. On actual boards there are a total of 20 pins in each direction for data, control, and test signals. There are two bases: one for each direction to maximize the signal integrity and to utilize digitally controlled impedance-matching (DCI) feature offered in VirtexII FPGAs. In case of dimensional constraints, the user can reduce the number of interface lines between the two boards. The minimum number of lines for each direction is 12. They are TXEN as the data valid, TXDATA [8:0] as the 8-bit data flowing into the test FPGA, and TXCLK together in phase with the TXDATA, and two bits of TXCONTROL for 4 possible conditions. At

higher frequencies (80 MHz and up), transmission line delay becomes a factor.

Several approaches are used to synchronize data between the two boards by compensating for the transmission line (cable) delay. In the first approach, by using the same clock on both boards, the driver board sends the TXCLK together with its data and the test board then clocks all its internal logic with this TXCLK. The test FPGA uses the falling edge of the TXCLK to clock all its logics. So when the test FPGA sends the data back to the driver FPGA, it will not send back the clock, and the driver FPGA on the receiving end has to phase-shift its clock to be able to correctly clock in the data to compensate for the cable wire delay. If the test FPGA uses the falling edge of the TXCLK, the driver FPGA still uses its original clock.

In the second approach, two clock domains may exist inside the driver FPGA and only one in the test FPGA. This method is suitable if the operating frequency is relatively low, since it is easier and safer to say one fixed phase shift value will work for any board under any temperature variations. But for higher-speed clock rates, the window is a lot smaller and using two-clock domains is more complex.

Using only one clock domain for the entire system is simpler. However, one has to watch and compensate for the delay introduced by the long cables. At clock rates over 25 MHz, the DCM feature of VirtexII FPGA becomes available. This feature offers the capability of deskewed clocks with proper feedback.

In the given test bed, all the cables are as short as possible having exactly the same length, to simplify single-clock-domain utilization for both boards. The design can effectively use the driver board to clock out data 'early' enough to compensate for the wire delay so that when the test FPGA gets the data, it will satisfy all the setup and hold time. The fan-out traces for UUT FPGA are almost the same length of about 4 inches. The test system offers a choice of four independent oscillators; two commercial on the driver board and two MIL Spec on the UUT board.

The challenge of using multiple clock domains deserves more attention. Considering that in the design, the feedback path is exactly the same length as the source path, it is possible for two FPGAs to operate using totally independent clocks. That means that the two clocks have no phase relationship whatsoever, even if they are relatively the same in frequencies. This choice leads to a simple PCB solution at the cost of more complex interface designs. The driver FPGA will send its TXCLK along with its data to the test FPGA, which will only use the TXCLK to clock in data at its IOB. Then it will use an asynchronous FIFO to convert the data into its own clock domain. When the test FPGA clocks out test results back to the driver FPGA, it does the same, so that the driver FPGA will have another asynchronous FIFO to reverse-convert the data into its own clock domain. This implementation works with any clock frequencies. However the flow control issue remains since over time the differences of the two different clocks will build up to cause one side to starve and the other side to overflow. So the interface logic has to be robust enough to tolerate such conditions. For the first release of the software and hardware, this approach was used.

In order to drive a good clock signal out of the IOB, not just any IOB can be used since their delays cannot be guaranteed. The signal often becomes very distorted. Xilinx DDR IO resolves this issue by 'regenerating' the clock using two flip-flops clocked by two clocks 180 degrees out of phase. One flip-flop will output a logic "one" and the other will output a logic "zero." This way it contains exactly the same delay timing with respect to any data clocked by the output flip-flop on other regular IOBs. This is widely used for high-speed off-chip interconnection systems.

5.0 Software Building Blocks

The rad-hard GUI software is a Windows-based application designed to control the hardware. The application has the ability to connect to the driver board, download the test FPGA code, download the test FPGA vectors, start testing, stop testing, collect results, and log all actions it takes. This is all done through a graphical interface. The software is developed on Windows 2000 using Microsoft's Visual Studio.NET.

The graphical interface consists of several buttons and a display window. The buttons that control the hardware work as a cascade, i.e., only the next action is available to the user, the other buttons are grayed out. Each press of a button will cause the action to be taken, a message displayed in the window, a message stored in the log, and the next button to be enabled.

The software itself is written in several layers. The topmost is the graphical interface, which is spread across several files. This layer controls the buttons and all OS level (i.e., file handling and memory allocation) issues. This is done through NET's

Forms capabilities. The interface components were dragged and dropped into place. From there, handlers were written to act when a specific button is pushed. This replaces the original command line. Other interfaces to the lower levels of the software can also be written.

The next layer is the driver layer. This is where the platform-specific commands belong. Functions like reading a specific register, downloading the test FPGA, or collecting test results are in driver.cpp. This layer is platform independent and can be moved from system to system with different interfaces. At this layer, calls to the logging system are allowed but printing to the screen or trying to access the graphical interface is not.

As far as the driver layer is concerned, there are only three 8-bit registers. They are control, status, and data registers, each with a unique address such as 0x01, 0x02, and 0x03. The parallel interface will facilitate the EPP mode specification, a typical bi-directional extension to the original printer port protocols. An indirect address scheme is used in order to do any transactions. In other words, two cycles of bus transaction are required to do any access to the three registers.

The next layer is the parallel layer, which is in parallel.cpp. This is where the details of talking to the parallel port are located. It is mostly a wrapper for the calls to the WinIO library. Replacing the bodies of PAR_ReadByte and PAR_WriteByte with calls to other types of hardware will permit moving the software from the parallel port to any other type of communication medium.

The bottom layer is a third party library, WinIO, which handles the actual details of communicating with the hardware. As with the Windows NT line of operating systems, direct hardware access is no longer permitted. A user space module needs to communicate with a kernel module in order to grant permission to read and write the hardware. WinIO provides this functionality for the project.

The log.cpp file controls the logging functionality. When the program starts, it will open and append all new information to the radhard.log file. If it is unable to successfully open this file, the program will not run. In this log is a time stamp with the date, hour, minute, seconds for each item stored. The time stamp is from the time of the operating system. This small part of the code is also not portable to other operating systems. There is no limit to the size of this file; the software just keeps appending to it.

6.0 Design Philosophy

The concept of the design reflects the features that were specified by Jet Propulsion Laboratory (JPL). The platform consists of the UUT (VirtexII 3000) placed in the socket on the burn-in (tester) board. The board is then positioned in the temperature chamber where the UUT is tested at temperatures from –55°C to + 125°C. The programmer/controller board controls the test.

The programmer/controller board and the burn-in board are connected to each other through the opening in the chamber by a total of 190 individual (3-foot, 50-ohm, 32 AWG) coaxial cables with high-temperature Teflon isolation. The 190 cables are composed of 76 input lines, 76 output lines, and 38 dedicated programs download and control lines. Extra lines for clocks and controls may be added if needed.

The entire design is powered by a low-noise, 10-Amp linear power supply with 1.5-, 3.3-, and 5.0- volt DC outputs.

The final component of the platform is a personal computer running Windows 2000 OS. The purpose of the personal computer is to provide nonvolatile storage space for configuration files, test vectors, and test results.

6.1 Burn-In Board Description

The main philosophy in the design of the burn-in board is maximum reliability and maximum signal integrity. This is achieved by minimizing the number of components on the board while preserving desired board versatility.

Since the power consumption is dynamic, it is a non-trivial task to ensure stable supply voltages at the device pins and to minimize ground differentials. The burn-in board consists of 14 layers of polyimide assembly with an uninterrupted ground plane, one plane for core supply V_{CCINT} (1.5 V) and one plane for V_{CCAUX} (3.3 V). V_{CCO} (3.3 V) is distributed on wide signal traces with sufficient bypass capacitors. Fast-changing Icc transitions are supplied by local decoupling capacitors placed very close (physically) to the Vcc device pins. These capacitors must have sufficient capacitance to supply Icc for a few ns, and must have low intrinsic resistance and inductance. 0.1 µF NPO ceramic, surface-mounted capacitors can supply 1 A for 2 ns with a 20-mV drop. There are 104 of these capacitors on the board. A half-inch or 10 mm trace represents an inductance of several nano-henries, defeating the purpose of the

decoupling capacitor. To compensate for trace inductance, there are 24 of the 4.7 µF and 2.2 µF tantalum capacitors positioned in various locations.

There are also four 100-uF power-supply decoupling electrolytic capacitors in order to supply even more current for a portion of the supply-switching period. (multiple smaller capacitors in parallel offer lower resistance and inductance than any large single capacitor.)

There will be a need to test more than one device. Since using multiple soldering processes for replacement of the UUT will result in rapid destruction of the test board or/and the tested device itself, the UUT must be placed on a socket. The burn-in BGA-728 socket was placed in the "center" of the board. The socket will allow for easy, solder-less replacement of the UUT whenever desired. The socket is discussed in more detail below. The socket is fanned out into six 76 pins, 50-ohm Mictor (or equivalent) connectors. These connectors are used for general I/O interface providing accessibility for the maximum possible number of I/Os of a given FPGA.

One more 38-pin Mictor connector is dedicated to the configuration / read-back function. All of the above constitute the necessary components to make the burn-in board functional. All the components are military grade.

Adding versatility to the board, there are two sockets for dedicated oscillators. These sockets will allow easy replacement of oscillators if different frequency sources are desired. The reason for having two sockets is to provide multiple and simultaneous sources of clocks and to enhance test resources during the test and exercise of the DCM feature of the FPGA. The disadvantage of sockets is that they require via-holes, which in turn may have some negative effect on signal integrity at very high frequencies. If on-board oscillators are not installed, the test board can use one of three oscillators located on the programmer/controller board. There are a few more clock combinations available to the user. All the oscillators feature tri-state outputs, which allow software control over the clocks. Oscillator selection is done by a jumper located close to one of the clock inputs of the FPGA. Optional on-board oscillators are high reliability, military grade manufactured, per order, by the Statec Company. The operating temperature range is 55°C ~ +150°C.

7.0 Additional work

There are still several features that could be added to improve the testing platform. The first release does not have external memory support due to PCB and FPGA design complications. A large number of test vectors are needed to truly test the FPGA. In the current version, these test vectors have to be broken down into several smaller test sets. This may introduce idling gaps between tests that might not be desirable. An extra 256 Kbytes of SSRAM or even 32 Mbytes of SDRAM will benefit the testing even though the verification design will be a lot more complex due to longer turn-around time of the RAM access time compared to that of the internal block RAM.

Another useful addition would be to make the driver FPGA capable of generating large pseudo-random data patterns on a clock-by-clock basis. This way the software does not have to initialize the test vector memory. When it comes to verification, comparing the output with what was generated is not implemented since there would be too many vectors to buffer. In the current design version, to do this would require a copy of the test FPGA to be stored within the driver FPGA and apply the test vectors to both the test FPGA and the copy of the test FPGA in the driver FPGA. Thus the test results from both FPGAs should be identical. Once any discrepancy occurs among the outputs, then it is known for sure that the error takes place in the UUT. The chance that all the identical outputs yield the same erroneous result is very slim. The objective is to loop-back all the IOs in the test FPGA and configure half of the IOs as inputs and the other half as outputs. A voltage level shifter in the loopback path can then be added in. Thus any patterns can be driven out of the outputs and inspect the inputs to see whether the right data has successfully been latched. The various IO standards can be tested this way and the voltage level shifter helps stressing the tolerance of the chip.

8.0 Conclusion

A fully programmable hardware/software platform for comprehensive static/dynamic burn-in test, at speed test, and SEU test of the VirtexII 3000 FPGA has been developed. The platform is capable of testing any possible configuration of the FPGA and any associated performance parameters. It also allows complete or partial re-programming of the FPGA and verification of the program by using read-back followed by dynamic test. Designers have full control over which functional elements of the FPGA to stress. Another benefit of this platform is that it

allows collecting information on elevation of the junction temperature as a function of gate utilization, operating frequency, and functionality. A software tool has been implemented to demonstrate the various features of the system. The software consists of three major parts: the parallel interface driver, main system procedure, and a GUI.

Two PCB boards (driver board and UUT board) have been fabricated with a specially designed power supply. The system has run for several weeks at room temperature, without a single error. At a temperature of 85 degrees C, hundreds of test vectors have been applied. Full burn-in at 125 degrees C, and SEU testing is currently planned for a military version FPGA.

9.0 Acknowledgements

This research was carried out, in part, at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA).

The authors wish to thank Gary Swift of JPL for his valuable comments and suggestions at the inception of this project. We would also like to thank Xilinx for their cooperation and material support.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

10.0 Bibliography

1. *Xilinx* VirtexII *TM DataSheets, (DS031.PDF),* http://www.xilinx.com

2. *Xilinx EPROM Application notes (XC18V0X.PDF),* http://www.xilinx.com

3. *Xilinx FPGA Downloading/Readback Applicaton notes (xapp138.PDF),* http://www.xilinx.com

4. *Precision Interconnect Blue Ribbon Solution*, http://www.precisionint.com/